Data-Driven Abstraction of Monotone Systems
Anas Makdesi, Antoine Girard, Laurent Fribourg

To cite this version:
Anas Makdesi, Antoine Girard, Laurent Fribourg. Data-Driven Abstraction of Monotone Systems. Learning for Dynamics and Control Conference, Jun 2021, Zurich, Switzerland. hal-03216643

HAL Id: hal-03216643
https://hal.science/hal-03216643
Submitted on 4 May 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Data-Driven Abstraction of Monotone Systems *

Anas Makdesi
ANAS.MAKDESI@L2S.CENTRALESUPELEC.FR

Antoine Girard
ANTOINE.GIRARD@L2S.CENTRALESUPELEC.FR

Laurent Fribourg
FRIBOURG@LSV.FR

Université Paris-Saclay, CNRS, CentraleSupélec
Laboratoire des Signaux et Systèmes
91190, Gif-sur-Yvette, France

Université Paris-Saclay, CNRS, ENS Paris-Saclay
Laboratoire Méthodes Formelles
91190, Gif-sur-Yvette, France

Abstract

In this paper, we introduce an approach for the data-driven abstraction of monotone dynamical systems. First, we introduce a set-valued simulating map, which over-approximates the dynamics of an unknown monotone system, using only a set of transitions generated by it. We establish the minimality of the introduced simulating map. Then, we show that the system, with this map as its transition relation, is equivalent (in the sense of alternating bisimulation) to a finite-state system. This equivalence enables the use of well-established symbolic control techniques to synthesize controllers. We show the effectiveness of the approach on a safety controller synthesis problem.

Keywords: Monotone transition systems, data-driven control, symbolic control, abstraction.

1. Introduction

Abstraction-based synthesis techniques have been gaining momentum in recent years (see e.g. Tabuada (2009), Belta et al. (2017) and the references therein). Using those techniques, we can build a finite-state approximation of a dynamical system, called “symbolic abstraction”. Then, controllers for the finite-state approximation are synthesized using algorithmic approaches adopted from the fields of supervisory control (Ramadge and Wonham (1987)), or formal methods (Bloem et al. (2012)). The synthesized controllers can address broad types of specifications such as safety and reachability (Girard (2012)), behaviors described by linear temporal logic (Baier and Katoen (2008), Belta et al. (2017)) or finite automata (Pola and Di Benedetto (2019), Sinyakov and Girard (2019)). Those specifications go beyond the traditional properties, such as stability, to describe a desired behavior of the controlled system over time. The synthesized controllers can then be refined to controllers for the original system if some behavioral relationship relates the symbolic abstraction with the original system. In such cases, the refined controllers are “correct by construction”. An example of such a relation is the alternating (bi)-simulation relation (Tabuada (2009)), or the feedback refinement relation (Reissig et al. (2016)).

Although symbolic control is essentially a model-driven technique, we present in this paper a data-driven approach to calculate the symbolic abstraction. Starting from a set of transitions...
generated from an unknown monotone system, we find a simulating map over-approximating the
dynamics of any system capable of producing these transitions. This simulating map is minimal
in the sense that it is included in any other map doing the same work. We then calculate a sym-
bolic abstraction of the system represented by this simulating map, and we prove the existence of
alternating bisimulation relation between them.

Similar work to compute symbolic abstractions from data has been done in (Sadraa and
Belta (2018), Hashimoto et al. (2020)), but while they require bounds on the system continuity
in a Lipschitz sense, our method does not have this requirement. We rather study the class of
monotone systems and build our data-driven abstraction based only on the monotonicity property
of the system.

Monotone systems can be found in many applications such as power networks (Zonetti et al.
(2019)), traffic networks (Kim et al. (2017)), and biomedical systems (Angeli and Sontag (2003)).
Meyer et al. (2015) applied symbolic control methods on a monotone temperature regulation system
by only sampling the system dynamics on a given grid of states and inputs. Saoud et al. (2019) pre-
sented an efficient symbolic control synthesis technique for monotone systems, which was applied
on a vehicle platooning problem.

The paper is organized as follows. In Section 2, some required preliminaries are provided.
In Section 3, we describe an approach to compute a set-valued map that over-approximates the
true system dynamics. In Section 4, we show that the dynamical system induced by this map is
alternatingly bisimilar to a finite-state abstraction. Finally, In Section 5, we show the effectiveness
of our approach by studying a cruise control problem.

2. Preliminaries

In this section, we define some notations and introduce the class of monotone dynamical systems
under consideration in this paper.

2.1. Notation

\( \mathbb{R}, \mathbb{R}_0^+, \text{ and } \mathbb{N} \) denote the sets of real, non-negative real, and natural numbers, respectively. The
empty set is denoted by \( \emptyset \). Given a set \( X \), we denote \( 2^X \) to the set of subsets of \( X \). We define the
partial order \( \preceq \) on \( \mathbb{R}^n \) as follows. Let \( x = (x_1, \ldots, x_n) \), \( x' = (x'_1, \ldots, x'_n) \) in \( \mathbb{R}^n \), we say that:

\[ x \preceq x' :\iff (x_i \leq x'_i, \forall i = 1, \ldots, n). \]

If \( x \preceq x' \) then \( x' \succeq x \). A relation \( R \subseteq X \times Y \) is identified with the set-valued map \( R : X \rightarrow 2^Y \)
where \( R(x) = \{ y \in Y | (x, y) \in R \} \). The domain of \( R \) is \( \text{dom}(R) = \{ x \in X \mid R(x) \neq \emptyset \} \). The
Cartesian product of the indexed sets \( (X_i)_{i \in I} \) is denoted by \( \prod_{i \in I} X_i \).

2.2. Monotone systems

First, let us consider a discrete-time dynamical system of the form:

\[ x^+ \in F(x, u), x \in X, u \in U \quad (1) \]

where \( F : X \times U \rightarrow 2^X \) is a nonempty set-valued map, \( X \subseteq \mathbb{R}^n, U \subseteq \mathbb{R}^m \). We say that \( F \) is
deterministic if for all \( x \in X, u \in U, F(x, u) \) is a singleton (in this case we identify the set-valued
map $F$ with the classical map $F : X \times U \to X$), otherwise it is said to be non-deterministic. We consider a partial order on the set of states $\mathbb{R}^n$ and the set of inputs $\mathbb{R}^m$.

**Definition 1** $F$ is monotone if for all $x, x' \in X, u, u' \in U$ with $x \preceq x', u \preceq u'$,

$$\forall y \in F(x, u), \exists y' \in F(x', u'), y \preceq y', \text{ and }$$

$$\forall y' \in F(x', u'), \exists y \in F(x, u), y \preceq y'.$$

![Figure 1: The set-valued image of two points, one is larger than the other and the map is monotone](image)

Figure 1 shows a set-valued image of two points. Under the previous monotonicity definition, we do not require all the points in the image of $x'$ to be larger than the points in the image of $x$. In the case where $F$ is deterministic, Definition 1 becomes:

$$x, x' \in X, u, u' \in U, x \preceq x', u \preceq u' \implies F(x, u) \preceq F(x', u')$$

which resembles the ordinary definition of monotone functions.

3. **Learning monotone systems from data**

In this section, we present a data-driven approach to compute an optimal and guaranteed over-approximation of an unknown monotone map, using only a given set of transitions generated by it.

3.1. **Formulation**

We are given a set of transitions $\mathcal{D} = \{(x_k, u_k, x'_k) | k = 1, \ldots, N\}$, this data has been generated by an unknown deterministic and monotone map $\tilde{F} : X \times U \to X$.

**Definition 2** A deterministic map $F : X \times U \to X$ is consistent with data $\mathcal{D}$ if:

- $F$ is monotone;
- $F(x_k, u_k) = x'_k$, $k = 1, \ldots, N$.

Let us remark that the true unknown map $\tilde{F}$ is consistent with the data. However, there are generally an infinity of maps consistent with the data $\mathcal{D}$. We denote the set of consistent maps by $C(\mathcal{D})$. We aim at computing a set-valued map that captures all of them.
Figure 2: Data points (left) and their successors (right). On the left, the point $x$ is smaller than the points $K^+ = \{4, 5, 7\}$ and bigger than the points $K^- = \{1, 2\}$. On the right, the set-valued map $\tilde{G}(x, u)$ is represented by the purple area.

Definition 3 A set-valued map $G : X \times U \rightarrow 2^X$ simulates $D$ if:

- It is monotone;
- For all $F \in C(D)$, for all $x \in X$, $u \in U$, $F(x, u) \in G(x, u)$.

A map $G$, simulating $D$, captures the true unknown map $\tilde{F}$: for all $x \in X$, $u \in U$, $\tilde{F}(x, u) \in G(x, u)$. There are generally an infinity of maps that simulates $D$. We denote the set of simulating maps by $S(D)$.

Definition 4 $\tilde{G} : X \times U \rightarrow 2^X$ is the minimal simulating map of $D$ if:

- It simulates $D$;
- For all $G \in S(D)$, for all $x \in X$, $u \in U$, $\tilde{G}(x, u) \subseteq G(x, u)$.

The following proposition provides the expression of the minimal simulating map. Figure 2 illustrate the proposed map.

Proposition 5 There exists a unique minimal simulating map of $D$ given by

$$\tilde{G}(x, u) = \left( \bigcap_{k \in K^- (x, u)} \{ y \in X \mid x_k' \leq y \} \right) \cap \left( \bigcap_{k \in K^+ (x, u)} \{ y \in X \mid y \leq x_k' \} \right)$$

where

$$K^- (x, u) = \{ k \mid x_k \leq x, u_k \leq u \}, \quad K^+ (x, u) = \{ k \mid x_k \leq x, u_k \leq u \}. $$

Proof Let us start with existence. Let $x, x^* \in X$, $u, u^* \in U$ with $x \leq x^*, u \leq u^*$. We define

$$y = \left( \max_{k \in K^- (x, u)} x_{k,1}', \ldots, \max_{k \in K^- (x, u)} x_{k,n}' \right),$$

where $x_{k,i}'$ is the $i$th component of the $k$th data point’s successor. The point $y$ represent the smallest point in the image $\tilde{G}(x, u)$ because it is the smallest point in the intersection $\bigcap_{k \in K^- (x, u)} \{ y \in \}$.
$X \mid x_k' \leq y \}$. For example, in the case of Figure 2 the point $y$ is $(x'_{2,1}, x'_{1,2})$. $y$ belongs to $\tilde{G}(x, u)$ and because the partial order relation is transitive, we can say that $x_k \leq x \leq x^*$ and $u_k \leq u \leq u^*$ for all $k \in K^-(x, u)$, which means that the set $\bigcap_{k \in K^-(x, u)} \{ y \in X \mid x_k' \leq y \}$ is included in the set $\bigcap_{k \in K^-(x, u)} \{ y \in X \mid x_k \leq y \}$. Therefore, we now have:

$$\forall y^* \in \tilde{G}(x, u)^*, y^* \succeq y \in \tilde{G}(x, u) .$$

Let us also define the point

$$\overline{y^*} = ( \min_{k \in K^+(x^*, u^*)} x'_{k,1}, \ldots, \min_{k \in K^+(x^*, u^*)} x'_{k,n} ) .$$

$\overline{y}$ is largest point in $\tilde{G}(x^*, u^*)$ and we know that $x \preceq x^* \preceq x_k$ and $u \preceq u^* \preceq u_k$ for all $k \in K^+(x^*, u^*)$. In the same way as above we can say:

$$\forall y \in \tilde{G}(x, u), y \preceq \overline{y^*} \in \tilde{G}(x, u^*) .$$

From (3) and (2) $\tilde{G}$ is monotone.

For all $F \in C(D)$, for all $x \in X$, $u \in U$, $F$ is monotone which implies that $F(x, u) \succeq x_k'$ for all $k$ such that $k \in K^-(x, u)$ and $F(x, u) \preceq x_k'$ for all $k$ such that $k \in K^+(x, u)$. Therefore, $F(x, u) \subseteq \tilde{G}(x, u)$. Added to the fact that $\tilde{G}$ is monotone, we now have that $\tilde{G}$ simulates $D$.

Then, we want to prove that for all $G \in S(D)$, for all $x \in X$, $u \in U$, $G(x, u) \subseteq \tilde{G}(x, u)$, so let us assume that it is wrong and that there exists $x^* \in X$, $u^* \in U$ such that there is $y \in \tilde{G}(x^*, u^*)$ but $y \notin G(x^*, u^*)$. Let us consider the deterministic and monotone map $F$ satisfying $F(x_k, u_k) = x_k'$ for all $k = 1, \ldots, N$ and $F(x^*, u^*) = y$. Then, $F \in C(D)$ but $F(x^*, u^*) \notin G(x^*, u^*)$, which means that $G$ does not simulate $D$, and we reach a contradiction.

We now prove uniqueness. If $\tilde{G}_0$ and $\tilde{G}_1$ both are minimal simulating map, then $\tilde{G}_0(x, u) \subseteq \tilde{G}_1(x, u)$ and $\tilde{G}_1(x, u) \subseteq \tilde{G}_0(x, u)\) which mean $\tilde{G}_1(x, u) = \tilde{G}_0(x, u)$ for all $x \in X$, $u \in U$. \qed

It is worth noting that because $\tilde{G}(x, u)$ is the intersection of areas bounded from above and from below by the successors of the data points, adding new data points will make $\tilde{G}(x, u)$ more resembling to the true map $\tilde{F}$. This can be expressed in following property:

$$D \subseteq D' \Rightarrow \tilde{G}_{D'} \subseteq \tilde{G}_D$$

where $\tilde{G}_D$ and $\tilde{G}_{D'}$ denote the minimal simulating map of $D$ and $D'$ respectively.

### 3.2. Partitioning of the sets of states and inputs

Let us assume that sets of states and inputs are rectangular: $X = [a_1, b_1] \times \cdots \times [a_n, b_n]$ and $U = [c_1, d_1] \times \cdots \times [c_m, d_m]$. We can state the following result:

**Proposition 6** There exist finite rectangular partitions $(X_q)_{q \in Q}$, $(U_p)_{p \in P}$ of $X$ and $U$, a finite collection $(Y_{q,p})_{q \in Q, p \in P}$ of rectangular subsets of $X$ such that for all $x \in X_q$, $u \in U_p$, $\tilde{G}(x, u) = Y_{q,p}$.

**Proof** First, let us introduce the following grid: on each axis of the state space, e.g., axis $i$, $i \in \{1, \ldots, n\}$, we sort the corresponding components of all the data points (i.e., sort $x_{k,i}$, $k = 1, \ldots, N$), meaning that on each axis, we have $x_{k,1,i} \leq \cdots \leq x_{k,N,i}$. The sorted values on the axes of the state.
space define a grid. Inputs from the data set define a similar grid on the input space. Those grids will
enclose hyperrectangle cells. Later, we will see that the value of the map \( \tilde{G} \) will stay the same for all
the points inside a hyperrectangle in the state space, under any input inside a particular hyperrectangle
in the input state. To establish the finite rectangular partitions, we should also address the points
of the grids (boundaries between the cells). That is what we are doing next. To describe the partition
we will use two sets of symbols \( Q = Q' \cup Q'' \), where \( Q' = \{1, \ldots, N\} \) and \( Q'' = \{1, \ldots, N - 1\}^n \).
The first set of symbols \( Q' \) is dedicated to the data points, meaning:

\[
\forall q \in Q', X_q = \{x_q\}.
\]

A symbol \( q \in Q'' \) from the second set is a tuple \( q = (q_1, \ldots, q_n) \); It determines the hyperrectangle
position on each axis (i.e., \( q_i \) is an index that refers to one of the values in the list of sorted data
points’ \( i^{th} \) components), \( q \) allows us to define (the lower and upper values of the hyperrectangle)
\( x_q = (x_{k_1^{q_1}, 1}, \ldots, x_{k_n^{q_n}, n}) \) and \( \overline{x}_q = (x_{k_1^{q_1} + 1, 1}, \ldots, x_{k_n^{q_n} + 1, n}) \). Now we have all the parts to define
the partition:

\[
\forall q \in Q'', X_q = \left( \prod_{i=1}^{N} X_q^i \right) \setminus X_D
\]

where \( X_D = \bigcup_{k=1}^{N} \{x_k\} \), and \( X_q^i \) is defined as follows:

\[
X_q^i = \begin{cases} 
[x_{k_1^{q_1}, i}, x_{k_1^{q_1} + 1, i}] & \text{if } x_{k_1^{q_1}} \leq x_q \text{ and } \overline{x}_q \leq x_{k_1^{q_1} + 1} \\
[x_{k_1^{q_1}, i}, x_{k_1^{q_1} + 1, i}] & \text{if } x_{k_1^{q_1}} \leq x_q \text{ and } \overline{x}_q \leq x_{k_1^{q_1} + 1} \\
(x_{k_1^{q_1}, i}, x_{k_1^{q_1} + 1, i}) & \text{if } x_{k_1^{q_1}} \leq x_q \text{ and } x_q \not\leq x_{k_1^{q_1} + 1} \\
(x_{k_1^{q_1}, i}, x_{k_1^{q_1} + 1, i}) & \text{if } x_{k_1^{q_1}} \leq x_q \text{ and } x_q \not\leq x_{k_1^{q_1} + 1} \\
\end{cases}
\]

An example to illustrate the partition can be found shortly after the proof. After we define \( U_p \) in
the same way, we have under our definition for the partial order, for all \( x \in X_q \) and for all \( u \in U_p \),
\( K^-(x, u) \) and \( K^+(x, u) \) remain constant. Therefore, \( \tilde{G}(x, u) \) will have the same value \( \tilde{G}(x, u) = Y_{q,p} \)
for all \( x \) in \( X_q \) and for all \( u \) in \( U_p \). \( Y_{q,p} \) will either be a single point \( Y_{q,p} = \{x_k\} \) in the case
where \( X_q = \{x_k\} \) and \( U_p = \{u_k\} \) such that \( (x_k, u_k, x'_k) \in D \) or it will be define by the box \( Y_{q,p} = [\underline{y}, \overline{y}] \)
where \( \underline{y} = (\max_{k \in K^-}(x, u) x'_k, \ldots, \max_{k \in K^-}(x, u) x'_k) \) and \( \overline{y} = (\min_{k \in K^+}(x, u) x'_k, \ldots, \min_{k \in K^+}(x, u) x'_k) \)
either.

Figure 3 offers an illustrative example of a finite rectangular partition (in the state space) based
on four data points. We examine the cell which has the symbol \( q \in Q'' \) with \( q = (2, 1) \). We have
\( X_q = (x_{1,1,1} \times [x_{2,2,1}, x_{3,2,2}]) \). \( x_{1,1,1} \) is excluded because \( x_1 \not\leq x_q \), whereas \( x_{4,1,2}, x_{2,2,2} \), and \( x_{3,2,3} \) are
included because \( x_2 \preceq x_q, \overline{x}_q \preceq x_3, \) and \( \overline{x}_q \preceq x_4 \).

**Remark 7** In the following sections and in our numerical tests we will actually make the output
\( \tilde{G}(x, u) = Y_{q,p} \) for all the points \( x, u \) in the sets \( X_q = [x_{k_1^{q_1}, 1}, x_{k_1^{q_1} + 1, 1}] \times \cdots \times [x_{k_n^{q_n}, n}, x_{k_n^{q_n} + 1, n}] \)
and \( U_p = [u_{k_1^{p_1}, 1}, u_{k_1^{p_1} + 1, 1}] \times \cdots \times [u_{k_n^{p_n}, n}, u_{k_n^{p_n} + 1, n}] \), \( q \in Q = \{1, \ldots, N - 1\}^n, \)
\( p \in P = \{1, \ldots, N - 1\}^m \). With this new definition of \( \tilde{G}(x, u) \), \( \tilde{G} \) is not strictly speaking the minimal sim-
ulating map, but this new simulating map is easier to deal with, and it differs from the minimal simulation map only on a set of measure 0.
Figure 3: The partitioning of the state space based on the data points. Arrows indicate to which set, the boundaries between adjacent sets, belong.

In order to compute the sets \((X_q)_{q \in Q}\), \((U_p)_{p \in P}\) and \((Y_{q,p})_{q \in Q, p \in P}\) from the data set \(D\) we should follow these steps:

1. We sort the first component of our data points, then the second component, and so on till we reach the \(n_{th}\) component.
2. We do the same for input data till we reach the \(m_{th}\) component.
3. We partition the sets of states and inputs according to the sorted components.
4. We calculate \(\tilde{G}(x,u)\) for every set.

4. Bisimilar finite abstraction

In this section, we show the equivalence (in the sense of alternating bisimulation) between the system defined by the calculated minimal simulating map and a finite state transition system.

**Definition 8** A transition system \(\Sigma\) is a tuple \(\Sigma = (X, U, F, Y, H)\), where \(X\) is a set of states, \(U\) is a set of inputs, \(F : X \times U \rightarrow 2^X\) is a transition relation, \(Y\) is a set of outputs, and \(H : X \rightarrow Y\) is an output map.

\(\Sigma\) is finite if \(X\) and \(U\) are finite. Now let us consider two transition systems \(\Sigma_i = (X_i, U_i, F_i, Y_i, H_i), i = 1, 2\). We assume that the systems are observed over the same set of outputs \(Y = Y_1 = Y_2\).

**Definition 9** \(R \subseteq X_1 \times X_2\) is an alternating bisimulation relation between \(\Sigma_1\) and \(\Sigma_2\) if for all \((x_1, x_2) \in R\),

1. \(H_1(x_1) = H_2(x_2)\)
2. \(\forall u_1 \in U_1, \exists u_2 \in U_2, \forall x'_2 \in F_2(x_2, u_2), \exists x'_1 \in F_1(x_1, u_1)\) such that \((x'_1, x'_2) \in R\)
3. \( \forall u_2 \in U_2, \exists u_1 \in U_1, \forall x'_1 \in F_1(x_1, u_1), \exists x'_2 \in F_2(x_2, u_2) \) such that \( (x'_1, x'_2) \in R \)

\( \Sigma_1 \) and \( \Sigma_2 \) are alternatingly bisimilar if

1. \( \forall x_1 \in X_1, \exists x_2 \in X_2, \) such that \( (x_1, x_2) \in R \)

2. \( \forall x_2 \in X_2, \exists x_1 \in X_1, \) such that \( (x_1, x_2) \in R \)

In the following, let us consider that \( X_1 = X, U_1 = U, F_1 = \tilde{G} \) as defined in Remark 7. Let \( X_2 = Q, U_2 = P \) as in Remark 7, and let us consider the transition map \( F_2 : Q \times P \to 2^Q \) defined by

\[
F_2(q, p) = \{ q' \in Q \mid X_{q, p} \neq \emptyset \}.
\]

We consider the output set \( Y = Q \) and output maps given by \( H_1(x) = q \) if \( x \in X_q \) and \( H_2(q) = q \).

Let us remark that \( \Sigma_2 \) is a symbolic dynamical system (i.e. with finite set of states and inputs). In addition, let us assume that \( X = \bigcup_{q \in Q} X_q \) and \( U = \bigcup_{p \in P} U_p \).

**Theorem 10** \( \Sigma_1 \) and \( \Sigma_2 \) are alternatingly bisimilar and the alternating bisimulation relation is given by

\[
R = \{ (x, q) \in X \times Q \mid x \in X_q \}.
\]

**Proof** We will prove the statement for all \( x_1 \in X_1 \), there exists \( x_2 \in X_2 \), such that \( (x_1, x_2) \in R \). The proof of the second statement is similar, and is therefore omitted. For all \( x_1 \in X_1, x_1 \) should belong to one of the cells in the partition \( x_1 \in X_q \), so \( (x_1, x_2) \in R \) if we take \( x_2 = q \). Moreover, we will have:

1. From the definition of \( \Sigma_1 \) and \( \Sigma_2 \), we have \( H_1(x_1) = H_2(x_2) \);

2. For all \( u_1 \in U_1 \), there exists \( p \in P \) such that \( u_1 \in U_p \). So let us take \( u_2 = p \). For all \( x'_2 \in F_2(x_2, u_2) \), we are sure that there exists \( x'_1 \in F_1(x_1, u_1) = \tilde{G}(x_1, u_1) = Y_{q, p} \), such that \( x'_1 \in X_{x'_2} \), because we know, from the definition of \( F_2 \), that for any \( x'_2 \in F_2(x_2, u_2) \) we have \( X_{x'_2} \cap Y_{q, p} \neq \emptyset \), which means \( (x'_1, x'_2) \in R \);

3. For all \( u_2 = p \in P \), there exists \( u_1 \in U_p \), and for all \( x'_1 \in F_1(x_1, u_1) \), we know that \( F_1(x_1, u_1) = Y_{q, p} \), and \( Y_{q, p} \subseteq X \) which means that there exists \( x'_2 \) such that \( X_{x'_2} \cap Y_{q, p} \neq \emptyset \).

In another way, there exists \( x'_2 \in F_2(x_2, u_2) \) such that \( x'_1 \in X_{x'_2} \). In the end we have \( (x'_1, x'_2) \in R \).

As a consequence of the previous Theorem, it is optimal to use the abstraction provided by the finite-state transition system \( \Sigma_2 \) to synthesize a controller. This controller will work for all the systems consistent with the data; the finite-state transition system is alternatingly bisimilar to a system whose dynamics is simulating all of them. Not finding a controller using \( \Sigma_2 \) means that there is an uncontrollable system consistent with the data; any other map simulating the data will contain this uncontrollable system. Therefore, this is the best result we can find under the monotonicity condition.
5. Controller synthesis for safety specifications

To test the validity of the learnt abstraction, we want to use it to synthesize a controller for a safety problem, a commonly addressed problem in the field of symbolic control. Given a system $\Sigma = (X, U, F, Y, H)$ where $F = \tilde{G}$, and a safety specification $X^s \subseteq X$, we want to find a controller to keep the trajectories of this system inside the set $X^s$. This can be done by computing a safe controlled invariant defined as follows:

**Definition 11** A set $I \subseteq X$ is a safe controlled invariant for the system $\Sigma$ and the safe set $X^s$ if it satisfies:

- $I \subseteq X^s$
- $\forall x \in I$ and $\exists u \in U, F(x, u) \subseteq I$

We can use the symbolic abstraction presented in the previous section and the iterative algorithm described in Tabuada (2009) to find the maximal safe controlled invariant.

5.1. Numerical implementation (cruise control problem)

Let us consider a model with two vehicles moving in one lane on an infinite straight road. The leader has a fixed velocity $v_c$ while the follower is controllable. A discrete-time approximation of this model is given by equations:

$$
\begin{align*}
    d_{k+1} &= d_k + (v_k - v_c)T_0 \\
    v_{k+1} &= v_k + \alpha(u_k, v_k)T_0,
\end{align*}
$$

(4)

Here $u_k$ is the control input which is the torque applied to the wheels. $d_k$ is the signed distance between the vehicles.

$$
\alpha(u, v) = u - M^{-1}(f_0 + f_1v + f_2v^2);
$$

The vector of parameters $f = (f_0, f_1, f_2) \in \mathbb{R}_+^3$ describes the road friction and vehicle aerodynamics whose numerical values are taken from Nilsson et al. (2015): $f_0 = 52 N$, $f_1 = 1.2567 N s/m$, $f_2 = 0.4342 N s^2/m^2$. For the rest parameters we chose: $v_c = 20 m/s$, $M = 1370 kg$, $T_0 = 0.1 s$. This model is only used to generate the data. We wanted the follower car to stay in a distance between 10 m and 30 m from the leader, while maintaining a speed between 19 m/s and 21 m/s. Let us remark that over these ranges of values the system (4) is monotone. We sampled randomly 300 points inside this safe set. We sampled also 300 random input values. We calculated the successors from the model given by (4). Data points, successors, and inputs are shown in Figure 4.

Calculating the abstraction took 923 s to be finished (CPU: 2.9 GHz Intel Core i7, RAM: 8 Go 2133 MHz DDR34, Matlab R2020b). It took 341 s to calculate the invariant set. The red area in Figure 5 represents the invariant set that we found. It should be noted that the size of the abstraction grows polynomially with the number of data points. In the example there are two states $n_x = 2$ and one input $n_u = 1$. The number of sets in the partitioning is $N^{n_x+n_u} = N^3$ where $N$ is the number of data points. This makes using a big number of data points computationally difficult.
6. Conclusion

An algorithm for data-driven abstraction was introduced. To find the abstraction, we first calculated a map that minimally over-approximates the system dynamics. We showed that this abstraction contains all the information we can acquire from the data and the monotonicity condition. We demonstrated the effectiveness of our approach on a safety controller synthesis problem. The approach suffers from a scalability problem as the abstraction size grows polynomially with the number of data points.

As for the future, a way to deal with uncertainties and noises in the data points should be studied. Also, the scalability problem should be addressed maybe with the help of data aggregation techniques. The authors also want to investigate the use of data collected on-line to improve the calculated controller.
References


Vladimir Sinyakov and Antoine Girard. Formal controller synthesis from specifications given by

Paulo Tabuada. *Verification and control of hybrid systems: a symbolic approach*. Springer Science

Daniele Zonetti, Adnane Saoud, Antoine Girard, and Laurent Fribourg. A symbolic approach to
voltage stability and power sharing in time-varying dc microgrids. In *European Control Confer-